MANAGEMENT OF STORAGE SPACE FOR AN EMBEDDED DATABASE IN A SOFTWARE SYSTEM

Field

[0001] The present teachings relate to management of storage space for embedded databases in software systems.

Introduction

[0002] As the price of relational databases decreases and the performance increases, particularly in a personal computing environment, it is becoming increasingly popular to embed a relational database within software applications. In this setting, the relational database essentially becomes part of the application, and the software end-users are not aware of the presence of the relational database. In fact, it is often the goal and hope of application vendors that users do not know or do not need to know anything about the database. Customers purchasing this type of software application often lack the database resources or technical expertise to manage a database. But due to the intricacies of the relational database, the embedded database has to be managed somehow, especially related to its space. Because the application vendors do not know all of the computer configurations their applications are going to be installed upon, the common approach is that applications allocate a small amount of space when installed and make it auto extensible so that when a need arises to store more data, the database grows by itself (i.e., without user awareness and/or intervention) within a pre-set limit. The initial size of the database, and how big the database can grow, are all unknown to a user. There are a number of potential problems with this common approach; e.g.: (1) A user generally does not know what's going on with database space. When the database stops growing due to either reaching the preset limit or hard disk space max out, it's often too late, which can result in loss of data; and (2) Even if a user knows that the database is almost full from his/her experience with the application, there is generally no convenient way for him/her to allocate more space to the embedded database.

Brief Description of Figures

[0003] Figure 1 illustrates a graphical user interface (GUI) displaying available disk space to a user, and enabling the user to allocate a selected amount of the available disk space for use by an embedded database, in accordance with various embodiments. In the example

shown, in the course of installing an application (here, GeneMapper(TM) software from Applied Biosystems; Foster City, CA) with an embedded database, a user has allocated 3GB in drive "E:" and 50GB in drive "F:"

[0004] Figure 2 illustrates a GUI associated with a database manager that monitors disk space usage while an application is running, in accordance with various embodiments. The database manager can tell a user how much disk space is used and/or how much remains available. In the example shown, the database manager provides a disk allocation capability whereby a user can add more disk space to the application's embedded database.

[0005] Figure 3 is a block diagram that illustrates an example of a computer system, according to certain embodiments, upon which various embodiments of the present teachings can be implemented.

Description of Various Embodiments

[0006] Reference will now be made to various embodiments, examples of which are illustrated in the accompanying drawings. While the present teachings will be described in conjunction with various embodiments, it is not intended that the present teachings be limited to such embodiments. On the contrary, the present teachings are intended to cover various alternatives, modifications, and equivalents, as will be appreciated by those of skill in the art.

[0007] The present teachings relate to management of storage (e.g., disk) space for embedded databases in software systems.

[0008] The present teachings can be implemented in a computer system, such as a personal computer (PC), Macintosh, or similar system. In various embodiments, the present teachings are embodied, at least in part, in a graphical computing environment.

[0009] In various embodiments, the present teachings provide a software application comprising an embedded database and one or more storage-space-management tools. The database can comprise, for example, embedded SQL (SQL statements placed within an application program, sometimes referred to as a host program).

[0010] In various embodiments, the present teachings enable a user to (i) pre-allocate disk space for an embedded database; and/or, (ii) add more disk space to an application's embedded database. The former can be effected, in various embodiments, when the software application is installed, and the latter when the software is running (operating). One or both of these tasks can be accomplished via a graphical user interface (GUI) permitting and facilitating user interaction.

[0011] In various embodiments, for example, when the software application is installed, a pre-allocation scheme can be used to allow a user to pre-allocate disk space for the embedded database. The pre-allocation scheme can work, for example, in the following way: A search on the user's local machine can be performed and the available disk space presented to the user. The user can allocate a selected amount of the available disk space to the application. In some embodiments, for example, a pre-set default value can be filled-in (replaced) by the user. A non-limiting embodiment of a GUI permitting and facilitating such actions is shown in Figure 1. After the user fills-in the allocation, the installer creates the overall size of the embedded database. The database file(s) created in this scheme can be fixed in size. In this manner, the user knows exactly how big the database is when installed. Since a user often knows how much data he/she is going to produce (low/medium/high throughput), the approach provides an excellent solution.

[0012] In various embodiments, when the software is running (operating), a database manager can be used to monitor the disk space usage. The database manager can tell a user how much disk space is used and/or how much remains available. In some embodiments, this information can be categorized, for example, according to distinct types of application data so that a user can have fine control over which data needs more space. The database manager can provide an early warning mechanism to a user about the database space. In some embodiments, the database manager can also provide a disk allocation capability. A user can add more disk space to the application's embedded database, if he/she desires, to selected application data. This provides a dynamic solution for overrunning the application's storage problem.

[0013] Those skilled in the art will appreciate that any of a variety of user editable fields can be employed. For example, and without limitation, text fields, drop down lists, scroll bars or boxes, among others.

[0014] In some embodiments, when the space usage reaches a predetermined threshold, the system can auto adjust the maximal amount of space according to a predefined set of rules without user intervention. The adjustment can be done, for example, per data type stored in the database. Such adjustment can be recorded in the system so that it can be traced if needed.

[0015] The rules can be set, for example, by the developer (prior to sale or distribution; i.e., prior to reaching an end user), and optionally can be adjustable by the user for one or more parameters included in the rules. For example, for a rule like "Increment space by 5GB when the usage is reaching 80%", the "5GB" and "80%" values can be set upon release of

the software, and optionally can be adjusted (reset) by a user in certain user interface implementations.

[0016] Among other things, the pre-allocation scheme in the software installation process can allow a user to know exactly how large the data storage is. The disk allocation design in the database manager can, for example, allow a user to know the space usage and to allocate more space, if needed or desired.

[0017] The present teachings can be employed, for example, in life-sciences software applications (e.g., genomics; proteomics; etc.). For example, the present teachings can be embodied in a software system for (i) DNA and/or protein sequence analysis, (ii) polymorphism detection, (iii) allelic discrimination, (iv) gene expression analysis, (v) bio-analyte detection, (vi) comparative sequence analysis, and/or (vii) gene expression profiling, and the like. The present teachings can be incorporated in software applications such as, without limitation, GeneMapper software; BioTrekker software; SDS software; and/or SeqScape software (Applied Biosystems; Foster City, CA). Various embodiments, for example, contemplate databases storing, among other things, polynucleotide and/or protein data (e.g., sequence data; polymorphism data; mutation data; etc.). The present teachings (e.g., an embedded database and a space-management GUI at installation and/or while running) can be incorporated in software technology such as, without limitation, that disclosed in U.S. Patent Application Nos. 09/724910 (filed 28-Nov-2000), 09/911903 (filed 23-Jul-2001), 10/279746 (filed 23-Oct-2002), 10/293960 (filed 13-Nov-2002), 10/241751 (filed 09-Sep-2002); U.S. Provisional Patent Application No. 60/479332 (filed 18-Jun-2003); and U.S. Patent Nos. 5538897, 6229911, 6532462, 6567540, 6185561, 6484183; all of which are incorporated herein by reference.

[0018] As indicated above, the present teachings can be implemented in a computer system, such as a personal computer (PC), Macintosh, or similar system. Figure 3 is a block diagram that illustrates a computer system 500, according to various embodiments, upon which various embodiments of the present teachings may be implemented. Computer system 500 includes a bus 502 or other communication mechanism for communicating information, and a processor 504 coupled with bus 502 for processing information. Computer system 500 also includes a memory 506, which can be a random access memory (RAM) or other dynamic storage device, coupled to bus 502, and instructions to be executed by processor 504. Computer system 500 further includes a read only memory (ROM) 508 or other static storage device coupled to bus 502 for storing static information and instructions for processor 504. A storage

device 510 (such as, without limitation, a magnetic disk, optical disk, magnetic tape, or the like) is provided and coupled to bus 502 for storing information and instructions.

[0019] Computer system 500 can be coupled via bus 502 to a display 512, such as a cathode ray tube (CRT) or liquid crystal display (LCD), for displaying information to a computer user. An input device 514, including alphanumeric and other keys, is coupled to bus 502 for communicating information and command selections to processor 504. Another type of user input device is cursor control 516, such as a mouse, a trackball or cursor direction keys for communicating direction information and command selections to processor 504 and for controlling cursor movement on display 512. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

[0020] In operation, processor 504 can execute one or more sequences of one or more instructions contained in memory 506. Such instructions can be read into memory 506 from another computer-readable medium, such as storage device 510. Execution of the sequences of instructions contained in memory 506 causes processor 504 to perform the process states described herein. Alternatively hard-wired circuitry may be used in place of or in combination with software instructions to implement the present teachings. Thus implementations of the present teachings are not limited to any specific combination of hardware circuitry and software.

[0021] The term "computer-readable medium" as used herein refers to any media that participates in providing instructions to processor 504 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 510. Volatile media includes dynamic memory, such as memory 506. Transmission media includes coaxial cables, copper wire, and fiber optics, including the wires that comprise bus 502. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

[0022] Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD, DVD, and any other optical medium, punch cards, papertape, any other physical medium with patterns of holes, a RAM, PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave, or any other medium from which a computer can read.

[0023] Those having ordinary skill in the art will understand that many modifications, alternatives, and equivalents are possible. All such modifications, alternatives, and equivalents are intended to be encompassed herein.